Lehigh University
Lehigh Preserve

Theses and Dissertations

1988

Hypertext: a tool for system development documentation management /

Timothy B. Halton
Lehigh University

Follow this and additional works at: https://preserve.lehigh.edu/etd

 Part of the Industrial Engineering Commons

Recommended Citation
Halton, Timothy B., "Hypertext: a tool for system development documentation management /" (1988). *Theses and Dissertations*. 4873.
https://preserve.lehigh.edu/etd/4873

This Thesis is brought to you for free and open access by Lehigh Preserve. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Lehigh Preserve. For more information, please contact preserve@lehigh.edu.

Hypertext: A Tool for System Development Documentation Management

by

Timothy B. Halton

A Six Credit Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Industrial Engineering

Lehigh University

May, 1988

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

May 9 1946
(date)

Jesse C. Wigmore
Professor in Charge

H. E. Klaas
Chairman of Department

Table of Contents

Table of Contents

List of Figures

Abstract

Hypertext is a computer-supported medium for information in which many interlinked documents are displayed on a high resolution monitor. While the concept of hypertext is not new, the technology to make it effective is new. This paper outlines some of the features of hypertext, and then discusses in some detail the design issues in creating hypertext documents. Finally, a simple example illustrating the effectiveness of hypertext as an organizational tool for software development life cycle documentation is presented.

Chapter I

Introduction.

The word Hypertext was first coined by Ted Nelson back in 1965, when he was searching for a term to describe nonsequential writing using a computer. As a data retrieval method, the concept of hypertext was first described in an article entitled: "As We May Think" by Vannevar Bush.

Bush imagined a text storage system in which documents and their contents were indexed not alphabetically or numerically but by association. Each time a new document was created, logical links would be built that would connect it to other documents already in storage. Later those links could lead a researcher along paths to other related documents.

Hypertext is a different way of looking at information. It is an environment designed to help people formulate, structure, compare, and manage ideas. Hypertext is a way to link interrelated information so computer users could jump from topic to topic, find related subject areas, and extract only what they need from large quantities of information.

Traditional methods of viewing information from a computer are straightforward, linear in fashion: you read the first paragragh, then the second, and so on. Outline processors let you create documents with details that are hidden in a hierarchical tree structure.

Hypertext takes the next step, creating complex networks of information linked by pointers and cross references. With a true hypertext system, the user can read an entry, jump automatically to other related entries, find cross references to these entries, and easily jump back to your first document. A true hypertext system provides links between text, graphics, audio, video--any kind of piece of information that a computer can digitize and access.

One promising application for hypertext is in structuring on-line access to the large number of documents currently involved with the software development process: requirements definition, design, implementation, quality assurance and maintenance.

The purpose of the documentation base is to guide and direct software development and then archive the information so that it could be referenced during and subsequent software modification. Not only can hypertext provide more rapid and natural access to this documentation, but it can easily integrate the official documents with personal notes and memos.

Another new technology, CD ROM offers vast storage capacity on a personal computer. File directories can't effectively handle the large number of files CD ROM can hold. To keep data stored in batches, something more than just a faster version of file directories must be made available. Hypertext might be the answer.

Hypertext approaches data storage and retrieval associatively; a body of information is referenced within itself rather than to an outside directory. The key to implementing hypertext is the ability to constantly add new data and build new links with the information, which is what CD ROM cannot do.

Chapter II

The Character of Hypertext.

A natural question to ask is whether computerized document systems are as effective as paper in scanning large textual databases. Studies have shown that reading continous text from screens is approximately thirty percent slower than working from typewritten paper documents [Gould & Grischkowsky 84]. Although the computerized document systems are not directly comparable to paper books, the two serve a simular purpose. Both are used as sources of information, as learning devices, and as mechanisms for communication between people.

The essential idea of hypertext is the nonlinear or nonsequential organization of text. This section will discuss some important characteristics that have been demonstrated in existing hypertext systems.

Outlined below are features for an ideal hypertext system. The idea is to create a feel for using a hypertext system, both for writing and reading.

> The library or database is a network of textual or graphical nodes.
>
> Windows correspond to nodes in the library on a one-to-one basis, with each node identified by a name.

Windows can be repositioned, resized, closed and put aside as icons. Closing a window causes the window to disappear and the changes saved to the library. Clicking the mouse on the icon that has been put aside causes the window to open instantly.

Nodes may contain any number of links, which library. The link icon contains a short description to help define the contents of the node it points to. Clicking the link icon directs the system to find the referenced node and open a new window.

Hypermedia functionality is integrated into each application so that the amount of creating and traversing links are scattered with the actual creating and editing of documents.

The library may be browsed by following links and opening windows, by searching the network using a keyword, and by navigating through the document using a graphical browser.

The "browser" is an important component of a hypertext system. A browser displays some or all of the document as a graphical map. This road map can be used as a tool for finding one's way through the web of links. In addition, the browser display may be scanned or scrolled whenever the user has become "lost in space".

Editing Nodes

The most basic capability of a hypertext system is the ability to create and delete nodes and links, to modify the information contained in the nodes, and to modify the structure of the document. A complete version history of nodes and links may be maintained so that it

is possible to see any version of the document back to its beginning. Most hypertext systems include a facility that ensures that a link attached to an old version retains an attachment to a corresponding place in a new version.

The most desirable editing feature is a tool that provides direct manipulation of nodes. Direct manipulation is the ability to make well designed documents interactively which is very close to what-you-see-is-what-you-get (WYSIWYG) editing.

Authors and readers should have the same set of tools that permit them to browse through other nodes during the document preparation process and add notations to original links as they progress through a document. The wall between author and reader should then finally disappear.

It is important to consider not only the quality of the authoring tools, but also how these tools relate to one another. A fully-integrated hypertext system would permit reference copying and pasting of all data types. For example if a spreadsheet is copied and pasted into a hypertext document, it should retain the functionality of the spreadsheet program. That is, if the original spreadsheet is updated, all pasted copies

would have the option to be updated automatically by the system. For graphical images all editing functions would be activated when the cursor moves over the graphical image; and so on.

One method of electronic editing of documents when several authors are involved in working on the same document is to provide "markup tools". Authors can temporarily turn off the editing symbols and see a clean copy of the document [Brown 83]. Markup features allow an author to pick out the places where changes to a document have been made and distinguish between changes by different authors.

Nodes and Links

Links connect things together. For example:

> Links can connect the reference to another document to the document itself.
>
> Links can connect a comment or annotation to the text about which it is written.
>
> Links can indicate that some text is a subsection of some other piece of text, or other kinds of information.
>
> Links can connect two successive pieces of text, or a piece of text and all of its immediate successors.
>
> Links can connect entries in a table or figure to longer decriptions, or to other tables and figures.

The origination of the link acts as the reference. The source can be either a single point or a region of text. At the other end of the link, the destination, is referred to as the "referent". The destination can also be either a single point or a region of text. Most common implementations of links consist of a single point as the source and a region of text or a document as the destination.

The introduction of links into a hypertext system means that an additional set of tools are required to manage the use of links (ie., adding links, deleting links, renaming links, listing links, and so on).

There are generally three types of linking in hypertext: referential, organizational and implicit through the use of keywords. Referential links are the most common, where the source is a point or a region of text and the destination is a node. Organizational links connect a parent node with its children and form a tree structure within the network of nodes. Implicit linking connects a virtual set of nodes through a keyword search. Keyword searching is another technique for browsing large hypertext documents.

There are several ways in which the notion of links can be extended [Conklin 87]. Grouping links might be

useful for referring to several nodes with a single link.

Traversing the Document

A hypertext document is browsed by traversing links. Readers may restrict their attention to a single document by following only the links that serve to structure that particular document or readers may choose to follow diversions such as footnotes, references, or annotations that are linked to the document.

A hypertext reader following link after link in reading portions of documents may want to keep a trail of which links were followed. This trail allows other readers to follow the same path and makes it easier to resume reading a document after a digression.

Providing authors with a set of search and filtering tools, including the ability to search using a single criterion or to search using multiple criteria, a document system can filter the information for the reader. One can immediately imagine how useful this might be for filtering electronic mail. Other methods permit authors to associate keywords with any discrete piece of information, then readers can selectively view

the contents and eliminate material that is currently not relevant.

Using keywords and filters for finding specific information, hiding unnecessary detail, or customizing the display of the information illustrates how readers can use this system to personalize the presentation of information in ways that are not possible with paper media.

Hypertext Nodes

A hypertext document is meant to be viewed interactively. As the reader views a node, links may be followed or not at the discretion of the reader. If a link is followed, then the node at the other of the link is made visable so that it may be read. Providing a pictorial view of a document is an alternate way of selecting a node for reading.

Hypertext permits the writer to modularize ideas into units such that an individual idea can be referenced elsewhere and more detail could be offered to the reader. The sizing is at the discretion of the writer; however, there is nothing preventing the writer from storing a whole book of information into a single node. The impact on the reader due to the size of the node is not very well understood [Shasha 86].

Multi-User Access

A good hypertext system should provide a means for promoting the connection of ideas and communication among individual users. This means that the system should provide multiple users with access to the same document. At a minimum, multiple users should be allowed to create new links within a colleague's document.

To facilitate distribution of documents to a broader audience, hypertext systems should also provide tools for dissemination. They could include electronic mail features, networking, and automatic transfer of finished documents to typesetting equipment.

Advantages of Hypertext

Hypertext may offer new possibilities for creating and designing electronic documents. As the writer thinks of new ideas, he/she can develop them in their own nodes and link them to existing ideas, or leave them isolated if it's too early to make such associations. The hypertext functionality permits the movement from an unstructured network to the final document.

Hypertext also offers new possiblities for accessing large documents of information. A linear document can only be easily read in the order in which the text flows in the document. The advantage of nonlinear text is the ability to organize text in different ways depending on differing viewpoints.

Another advantage is that it is easy to suspend reading temporarily along one line of investigation while one looks into some other detail, and then return to complete the original investigation.

The interface for accessing and managing information is uniform across all possible kinds of information. Accessing and organizing graphics, images, and animation is done exactly the same way as accessing and organizing text (ie., by manipulating nodes and links).

Disadvantages of Hypertext

There are two types of problems that seem to be characteristics of hypertext. Along with the ability to organize information comes the problem of having to know where you are in the network and how to get to some other node that you think exists in the network. Hypertext offers more degrees of freedom, and therefore

a greater potential for the viewer to become lost or disoriented.

Reading hypertext documents tend to present the reader with a large number of choices about which links to follow and which links to leave alone. How do you decide if it is worth the digression?

The other problem with using hypertext is that it may become difficult to create, name, and keeptrack of links. When do you create a new link?

These questions and other design issues are addressed in the next chapter.

Chapter III

Design Issues in Building Hypertext Documents.

For hypertext systems to be effective, designers should understand how to locate facts to satisfy queries, or simply browse for information. Hypertext systems allow the user to traverse a complex network of nodes to find information quickly.

The primary goal of online queries is to retrieve information efficiently. Focusing on this goal yields efficient cost-effective performance. Browsing is an information seeking strategy for ill-defined problems. Hypertext systems encourage informal, content-oriented informaion seeking strategies for retrieving information.

Design decisions that affect information seeking in hypertext systems are related to defining access points, creating the user interface and providing good search strategies.

Defining Access Points

A key design question is what access points to define and how to link these access points? Users will expect

access points to be at least as rich as those available in books. Access points generally available in the printed book include the table of contents, indexes, glossary, physical pages, footnotes, and appendices. The page number is the primary pointer to a field location in a printed book.

Designers must decide how to offer access points and how much unrestrained jumping from point to point to allow. Links at every word are not desirable from the perspective of the user or of system performance. The trade-off in machine overhead and user overchoice must be weighed carefully.

User Interface

The introductory node is especially important. Generally readers will browse the introductory node first, so this node should reference as many key nodes as possible. There are different strategies for composing the introductory node.

One strategy is to make the introductory node rich in references. Following this strategy, the designer would make the introductory node a summary of the entire document. By scanning it, the reader can select one of many places to begin browsing.

A second strategy is to confine the introductory node to only a few key references. The idea would be to minimze the number of details which the reader should have to deal with to start browsing the information.

A third strategy is to design the introductory node as an outline. This technique could be extended to other nodes which could result in an extensive network of nodes. A particular node or document could appear in many places within the network, so readers can access it from many different points.

Its best to keep the contents of nodes short so that the reader need not deal with irrelevant information. Rather than include detailed information in a node the designer can reference it and create a separate node for it. This shields the reader from unnecessary details, but permits a path to the details if the reader feels it is relevant.

The interactive and flexible characteristics of hypertext require users to make more choices in searching for information. Improvements in screen readability through proper font design, text and background color and higher resolution would help the reader during the search process.

Search Strategies

Search features like Boolean connectives, string search, scope limits and truncation facilitate rapid access to information. Features for filtering such as frequency of occurrence per node or automatic thresholds should be available to alert the user to the effects of a particular search.

Flexibility leads to complexity. Designers must struggle with the issue of when to stop adding features, (ie., they face the law of diminishing returns).

The general problem of maximizing power and flexibility while minimizing complexity of use must always be addressed. Hypertext systems offer a clear advantage for finding facts, browsing information, and acquiring knowledge.

Chapter IV

Hypertext Implementations.

The history of hypertext is rich and varied, because hypertext is not so much a new idea as a natural use of the computer. After years of development the hypertext idea is finally showing up on personal computers. Hypertext has the potential to revolutionize the way written and graphic information is handled.

Hypertext is intended to bring publications into the computer age. Instead of reading straight through, the user may jump to related ideas. It's possible to read sequentially, but the user can also link up with other items, then go back and follow other leads. Reading with hypertext is open ended, the path being determined by the needs and interests of the reader.

The thesaurus on your desk may be considered a paper analogy of a hypertext system. It has no single beginning or end. Each time you use a thesaurus, you enter it at a different location based on the word that begins your search.

Another manual "hypertext system" is the use of 3 by 5 index cards. Index cards often reference each other, as well as being arranged in a particular order. One

advantage to index cards are their size which makes it easy to reorganize a set of cards when new information suggests a restructuring of notes. Their disadvantage is it might be difficult to locate a specific card if there are many of them. As will be discussed later, the 3 by 5 note card is a model for at least one Hypertext system (ie., the Macintosh Hypercard).

Memex

As mentioned above, Vannevar Bush may be credited with first conceiving a hypertext system in his article "As We May Think" [Bush 45], in which he visualized an electronic desk that would store full text of books, memos, articles, and other information. A device he called a "Memex".

Memex would give the author and the reader the ability to "build a trail through the maze of material available to him" [Bush 45].

Bush pointed out that our tools mimic earlier technologies. Using contemporary examples, word processors mimic typewriters, so they are constrained by the need to create a document defined by the width of a sheet of paper. Desktop publishing create more efficient ways to imitate typesetters. Outline processors imitate a book's table of contents.

Bush described a machine for browsing and making notes using an online text and graphics system. The Memex contained a very large library, as well as personal notes, photographs, and sketches. It also had several screens and a facility for linking between two points in the entire library. Bush admitted that many technological breakthroughs would be needed to make his Memex practical, but felt that it was an achievement worthy of the effort.

In the mid-sixties, Douglas Engelbart, working at Standford Research Institue developed the first implementation of a memex-like system called NLS (oNLine System) [Engelbart & English 68]. NLS ran on a mainframe and allowed researchers and other users to create sophisticated links between documents for applications as technical documentation and shared scientific research.

NLS has evolved over the years and is now called Augment. McDonnell-Douglas now markets Augment as a commercial network system. In developing NLS the emphasis was on creating a consistent environment for software engineers. The system includes many forms of computer-supported communication, both asynchronous and synchronous. There are also facilities for document

production and control, organizational and project management, and software engineering.

NLS also introduced the mouse, multiple windows and other innovations which are now common to the Apple Macintosh and Microsoft Windows environments.

Xanadu Project

The Xanadu Hypertext System [Nelson 80] is a unified system for storing and managing certain forms of information. It is built around four simple concepts:

Documents.

Multiple versions of documents.

Distributed Storage and distribued computations, involving networking between mainframe and microcomputers.

Isolation of storage functions from display functions.

A "document" is defined as a chunk of characters which functions in some way as a conceptual unit. It is a piece of text which some user has declared to be a single entity. It corresponds roughly to the notion of a file. A document may contain whatever the user decides to put in it. Possible documents include letters, books, chapters in books, memos, marginal notes, musical scores, reports, diagrams, pictures,

computer programs, maps -- whatever the user desires the document to be.

Each document stored in the system is assigned a unique identifier when it is created. The document identifier is a special number called a "tumbler". Tumblers are multi-part numbers similar to those used to number sections of technical manuals and to index books in the Dewey Decimal System.

Tumblers have the property that they can be expanded at any point, and it is therefore possible to implement a numbering system that never runs out of precision.

The Xanadu Hypertext system can manage multiple versions of a single document. Whenever a document is changed, the changes are recorded. The system remembers both the old and new versions and can retrieve either on command. Thus it is possible to retrieve a document in the state it was in at any point in time since its creation. In addition, the system can indicate the changes which caused two alternate versions to differ. In other words, a complete audit trail through the history of the document can be constructed.

Xanadu utilizes a simple set of editing commands by which it can be directed to make changes to a document. These are:

Insert characters at some point in the document.

Delete characters at some point in the document.

ion in the document to another.

Copy a span of characters from one location in the document to another.

A "link" is a logical connection between one piece of data and another. The Xanadu system supports a generalized linking facility. Any body of text in the entire system can be linked to any other body of text in the system. Links may connect text within a document or run between documents. The ability to maintain links across multiple versions of documents is one of the unique and powerful aspects of the Xanadu System.

Xanadu is designed to be a multi-user system, allowing independent users to access and manipulate the same body of information. Each user in the system has a unique user identifier. Each document in the system has an owner and the user identifier comprises one of the fields of the document identifier tumbler. By this means document ownership can be determined.

The owner of the document may grant other users permission to access it. The access privilege mechanism is as general as possible. One user may always edit another user's document to which he had access, but in doing so will have created a new version which is his own document.

The multi-user capability of Xanadu makes a number of useful activities possible. For example, a user can write marginal notes commenting on someone else's work. You write your comments in a document of your own and then link them to the other person's document. The other user may not even be aware of your comments. It is like writing in a library book without defacing the book to the eyes of others.

The Xanadu System is designed to be able to grow without having a significant impact on overall system performance. It does this by allowing the system to spread over as many networked machines as needed to support all users effectively. If a piece of information is requested at one node in the communications network but is not stored at that node, the software running at that node's computer sends a request thru the network to the machine in which the information is stored. This machine in turn sends back the requested data.

The system does not care what communications technology is used. No particular network topology is assumed, as long as paths exist between any nodes which might wish to communicate. The "backend" of the Xanadu System handles the storage, retrieval, versioning, linking, and editing of data. The backend is the portion which does all the "impossible things". The backend contains the proprietary components of the Xanadu system.

User interaction, the display of old data and the acceptance of new data is accomplished by the other portion of the system, the "frontend". The frontend worries about whether the data is text or pictures or music or whatever. The frontend decides how the data is to be formatted for display. The frontend knows how to interface with various input or output devices.

The frontend and the backend communicate using a standardized request and data transmission protocol. The frontend send commands and data to the backend which responds with other data and status information. The frontend will usually be on a separate computer which is connected to the backend by a communications line.

Frontends may be general purpose or special purpose. Different applications, different forms of data, different display and input devices and different user preferences can all be accommodated. For example, one frontend may function as an office automation frontend which handles text processing, electronic mail, and project management functions. A software development frontend may use Xanadu to manage source and object code and documentation.

Xanadu is built around a system of order that allows anyone to create personal information systems without confusion. Nothing is stored more than once, and everything already in the system is a boilerplate for new documents. That way it's easy to view the orgins of any item.

The backend of the Xanadu system has been implemented in UNIX. There is a "rough" frontend that runs on a SUN workstation.

Idea Processors

The goal of all idea processing tasks is to move from a chaotic collection of unrelated ideas to an integrated, orderly interpretation of these ideas. Idea processing

is a collection of three different phases: acquisition, analysis, and exposition.

Acquisition involves the capture of ideas and information from various sources ie., taking notes. Analysis involves discovering the significance of ideas. Exposition involves communicating ideas in the form of reports, talks, etc..

An idea processor is a tool for organizing blocks of text. Because text is a critical part of the final document, idea processors incorporate some type of text editor. Idea processors fall into two categories: outline organizers, and computerized index-card retrieval systems.

Outline organizers allow you to build outlines by means of headlines. Each headline can have any number of subheadings beneath it. Each subheading can have a paragraph. The subheadings can have more subheadings and paragraphs beneath them. You can expand a headline to see the subheadings and paragraphs beneath it, or you can collapse it to hide all subordinate levels. The ability to collapse or expand an outline at any level gives you many ways of viewing the ideas that make up your outline. This combined with the ability to move

headlines and insert new ones at any point, is the essence of idea processing.

An indexed cardfile system permits you to shuttle blocks of text quickly between the cardfile and your document. The cardfile resides in a hierarchical system of drawers and cabinets making it easy to relate to the real world. In the cardfile, you can peruse your indexed ideas to insert in your document. Or you can take a highlighted block of text from the main document and store it on a card for later use.

Notecards

Notecards is a computer environment designed to help people work with ideas [Halasz et al 87]. The system provides the user with a variety of tools for collecting, representing, managing, interrelating, and communicating ideas.

Notecards provide the user with a semantic network of electronic notecards interconnected by typed links. This network serves as a medium in which the user can represent collections of related ideas. The system provides the user with tools for displaying, modifying, manipulating, and navigating through this network.

A notecard is an electronic image of a 3x5 index card. Each notecard contains either a piece of text, a drawing, or a bitmap image. Each card has a title. Every notecard can be edited and there are various types of notecards.

Links are used to interconnect individual notecards into a network structure of related cards. Each link is a typed, directional connection between a source card and a destination card. The type of link is user defined specifying the nature of the relationship being represented. Its up to the user to organize a network of notecards.

A Browser is a notecard that contains a structural diagram of a network of notecards. Fileboxes are specialized cards that can be used to organize large collections of notecards. A Filebox is a card in which other cards, including other Fileboxes, can be filed. Fileboxes are designed to help users manage large networks of interlinked notecards by encouraging them to use hierarchical structures for efficient storage and retrieval of cards.

Navigation is the primary means for accessing information in Notecards. The user moves through the

network by following the links from card to card. In addition, Notecards provides a search facility that can locate cards by matching a user-supplied specification.

NoteCards was developed and used on Xerox D series Lisp machines, which are powerful enough workstations that they allow rapid creation and browsing of many hypertext nodes, as well as having a high resolution screen that allow node icons and links to be displayed in good resolution.

Hyperties

Hyperties is a software tool for organizing information [Shneiderman & Morariu 86]. Without the need for programming, you can link articles and illustration together so readers will be able to browse through them. The Hyperties System is broken down into two components: the Authoring System and the Browser.

The Authoring System permits you to create a database of articles and illustrations. Using the authoring system is very similar to a word processor. By adding text of articles to the database, you also specify the links or cross references to other articles or illustration. Hyperties automatically ties the articles

and illustrations together into a unified database and constructs an index to the entire database.

The Browser is used by readers to access the database of articles and illustrations. Readers can access complete articles, definition of terms, illustrations and cross references through Hyperties ability to create links. A link is a cross reference, an indication that more information on a particular word or phrase is available. Words or phrase that are links are highlighted on your computer display. If the computer is equipped with a touchscreen, readers can browse without the use of the keyboard at all. Hyperties runs on the IBM PC.

Intermedia

Intermedia is a tool designed to support both teaching and research in a university environment and is both an author's tool as well as a reader's tool [Yankelovich, Meyrowitz, & van Dam 85], [Garrett, Smith, & Meyrowitz 86]. The system runs on a network of UNIX based workstations and has five integrated applications: a text editor, a graphics editor, a scanned image viewer, a three-dimensional object viewer, and a timeline editor.

The text editor contains "style sheets", so the user can define a set of styles for a particular document and apply those styles to the text. With the graphics editor, users can create two dimensional illustrations. The scanned image viewer displays images that were entered through a digitizing scanner. These images can be cropped, copied, and pasted into text documents. The three dimensional object viewer converts files into three dimensional representations of the data. Users can manipulate these three dimensional objects by rotating them, zooming in or out, or hiding parts of the model.

The timeline editor provides interactive editing features for creating chronological timelines. As the user enters pairs of dates and labels, the application formats them on a vertical timeline. Similar to a charting package, the display of data is determined by a modifiable set of parameters.

In Intermedia, the hypermedia functionality is integrated into each application so that the actions of creating and traversing links can be scattered with the actions of creating and editing documents.

In any document, the user can specify a selection region as the starting point of the link. In any other document regardless of type, users can define another selection region as the end point of the link.

Intermedia allows users to create bidirectional links from a specified location in one document to a specified location in another document. These points are called blocks and the size of a block can range from the entire document to a single character on the document.

If the document containing links are deleted, the links are also deleted; however, the block markers at the other end of the link remain intact. Links and blocks are assigned descriptive properties. The link and block properties help manage the complexties within the Intermedia environment.

In Intermedia, block and link information are not stored within individual documents but are superimposed on them. "Webs" manage the block and link information, allowing one or more users to work with their own context. Intermedia incorporates a system of user access rights that helps manage multiple users sharing large bodies of connected material. The access scheme builds on protection mechanisms offered in most file

systems where user have either read or write permission. Intermedia adds annotation permission which allows users to add links to documents that they are not allowed to edit.

Macintosh Hypercard

Hypercard is a system for storing and cross-indexing information [Goodman 87]. The basic organizational unit is a card on a screen, like a rolodex card. Each card contains three features: fields, buttons, and graphics.

A field is a unit of textual information: a name, a number, or a complete document. A button is a live area that performs actions when clicked with a mouse. A button might dial a phone, perform calculations, or call up another card. Graphics can be diagrams, plans, illustrations, images scanned into the computer, or created with a paint program. Each card can contain all the graphics that will fit on the Macintosh screen at once.

The cards are arranged in "stacks" following the concept of the rolodex. You can easily flip from card to card. Hypercard's style on information management is less linear and more connective; its based on the way

we actually use information to perform real-world tasks.

Hypercard does not have a general mechanism for hierarchically structuring cards -- cards are grouped into stacks, but stacks cannot be grouped into stacks, so the "hierarchy" can only be one level deep.

Most computer programs are written in procedural languages. In a procedural language, the user simply creates a list of things for the computer to execute. The emphasis is placed on verbs, that is, on what the computer executes next.

In contrast, object oriented programming defines objects but does nothing until these objects receive messages. Object oriented programming gives knowledge to objects.

How does this work in Hypercard? As soon as a stack starts up, Hypercard "tells" the stack that it has started. When a user clicks a button, Hypercard sends a message and if the button has a routine to perform, that routine takes over.

These routines are written in Hypertalk. Hypertalk is an interpetive language that can be utilized to program

more advanced features of Hypercard. Hypertalk illustrates a technique called object oriented programming.

In full object-oriented languages, you can create not only objects, but classes of objects. Classes of objects are useful because they allow objects to share properties. Hypertalk consists of only five object classes: buttons, text fields, cards, backgrounds, and stacks. Hypertalk defines how each of them acts. Hypercard and Hypertalk are supplied as "standard equipment" on all Macintosh computers at present.

Guide

Guide is a Hypertext product for the IBM PC/AT and compatitable equipment. There are two distinct parts to using Guide--creating Guide documents, (known as Guidelines), and reading them. A Guideline looks short when it first appears on a screen. But as you pass the mouse over certain words in the text, your cursor changes to inform you of the presence of hidden layers of information "folded" under the visible top layer.

A Guideline can be a mixture of text and graphics. Certain words, phrases, or graphic objects can be

buttons that provide links to hidden text and graphics. Certain conventions are used to indicate the different types of buttons. For Example, boldface text indicates replacement buttons. Clicking on this button reveals hidden replacement text or graphics that are inserted after the button or displayed on top of it. Underlined text indicates a note button. Activating a note button, causes a small definition or memo to appear on the screen for as long as the mouse button is held down. Italics indicate a reference button, which opens up a new window to show a different Guideline document or branches to a different part of the Guidleline containing the reference button.

Text editing in Guide is performed the same way as with a screen editor. Guide adopts the approach that every character typed on the keyboard represents an insertion into the document. Readers can further tailor a document to their own needs by adding personal comments.

Guide permits resizing windows so that authors and readers alike can re-adjust the document layout for the most effective multiple window viewing and readabilty of documents. Documents can take any shape the user wants without being forced into a rigid hierarchy.

Guide is a tool for creating electronic documents to fit ideas. These documents can be multi-leveled, cross referenced, annotated, and animated. Instead of limiting the structure of a document to a traditional topic/sub-topic arrangement, a structure can be created that best fits the information.

Chapter V

A Support Tool for Software Development.

Much of the effort throughout the software development life cycle (SDLC) is concerned with the capture, representation, elaboration and presentation of design information (ie., information that is generated and collected during the software development process).

The amount of information that is relevant to the design process--the issues that arise during the design, alternatives, their resolution through decisions and, the rationale, work procedures and working relationships, notes and comments (including to-be-decided questions) and documentation should be represented effectively, so it will be accessible and useful by the design team.

The goal is to decrease the burden on the system designers by providing a tool to manage their data and their interdependencies.

Imperative to the success of any significant software project is the documentation of that project -- from the start of the project to the completion of the project. To aid in the generation and maintenance of good documentation, powerful yet easy to use tools

should exist that can flexibly manage a structured, centralized document respository.

A software design generally contains diverse elements ranging from structures as informal as natural language text to their structures as formal as statistical calculations. Inconsistent information arises in the early stages of the design because concepts are poorly understood. Furthermore, different concepts may be inconsistent with one another.

A system designer should be able to create a model representation for something that will eventually evolve into something more concrete. When designing a system, most items are related in various ways to many other items in the system. Such webs of information should be represented explicitly. Relationships can arise from both logical associations among objects and from constraints within the system. In the early stages of the design process, a design will change, thus creating the need for a tool to be flexible enough to deal with the evolving patterns of relationships.

The history of the system design process should be recorded as the design changes, and alternatives, including such things as revisions, and releases are defined. Information produced from the design process

represents the organizational glue for project documentation management. Hypertext can provide this organizational glue.

Hypertext as a Tool for Management Documentation

The following systems analysis and design tools are reasonable candidates for inclusion within hypertext.

Problem definition and requirements determination.

Data Flow Diagram illustrating the flow of information.

Data Dictionary.

Cross Reference.

Since it is always preferred to use concrete examples, this section will use the PC-based product Guide, as a medium of illustrating the potential of hypertext for the important use of applications. Guide assumes the system is running Microsoft windows. We will also assume a high-resolution (color) graphics monitor with a mouse.

The example described herein will introduce an approach that permits the capture and linking of information that will enhance the system designers'

ability to communicate ideas regarding current issues during the design process.

Using Guide, before we can browse through the information contained in the example -- following links, or creating them -- we must must first open a Guideline.

The introductory Guideline named CBM.GUI is illustrated in figure 1. Once the text is displayed we see that boldface text (ie., CBM Corporation, book-jobber, new management plans, operation considerably, see chart, and problems) are all replacement buttons. Underlined text like CBM is a note button. Italics (ie., Inventory Control System) represents a reference button which links to another guideline.

Figure 2 shows the results of clicking the book-jobber reference button. As you can see additional information is displayed. Following the Inventory Control System link, a different guideline is displayed. This is pictured in figure 3.

The word invoice and Data Flow Diagram are both reference buttons each linking to different guidelines. Traversing the Data Flow Diagram link displays a sketch of a simple data flow diagram.

The CBM Corporation.

Established twelve years ago, the CBM Corporation's business has been to act as a **book-jobber**.

While remaining profitable, CBM has come under pressure from competitors. As a result of a decision made by the board, the **new management plans** include expanding the **operation considerably**.

See Chart

This will create **problems** and also the need for an *Inventory Control System* of some sort.

Figure 1. The Introductory Guideline.

The CBM Corporation.

Established twelve years ago, the CBM Corporation's business has been to act as a ... (ie., Book Jobber receives orders from librarians for books about computers, ordering the books from the publisher at a discount, and filling the orders on receipt from the publisher).

While remaining profitable, CBM has come under pressure from competitors. As a result of a decision made by the board, the **new management plans** include expanding the **operation considerably**.

See Chart

This will create **problems** and also the need for an *Inventory Control System* of some sort.

Figure 2. Pushing the book-jobber button.

At the most general level, the proposed system will take book orders, check them against a files of books available, check against some file to verify the customer's credit, and cause the book(s) ordered to be sent out with an *invoice.*

This is illustrated by a logical *Data Flow Diagram*.

Figure 3. The Inventory Control System Guideline.

A logical Data Flow Diagram of the proposed Inventory Control System.

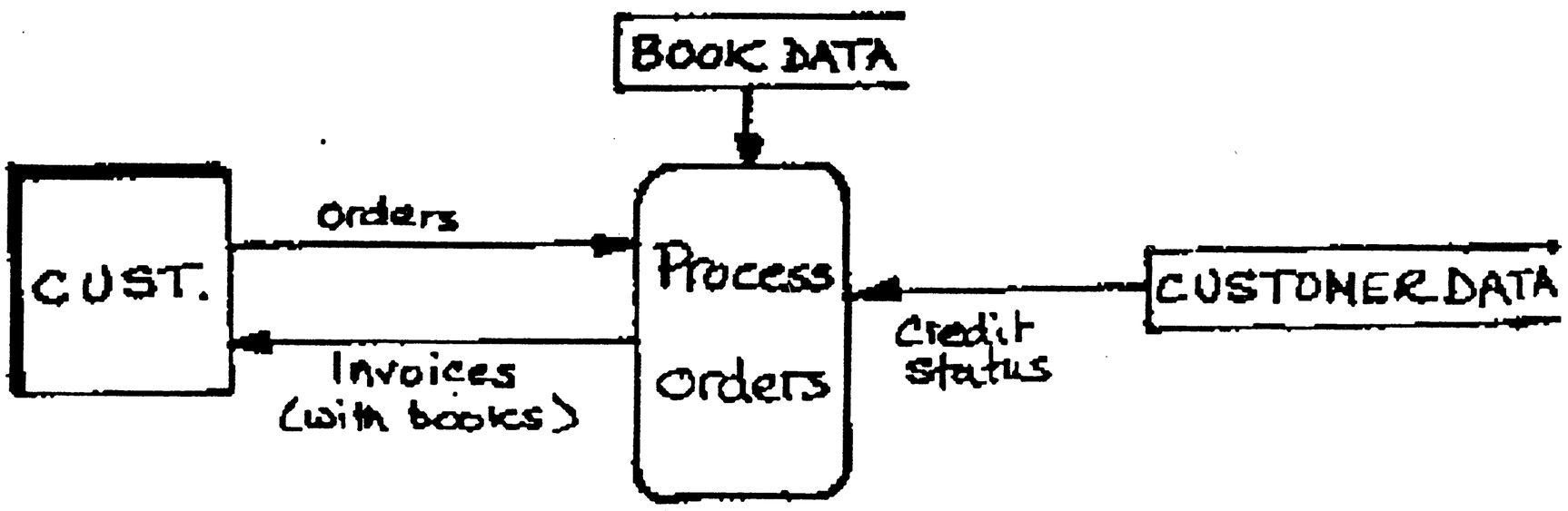


Figure 4. A Simple Data Flow Diagram.

This is shown in figure 4. The drawing was placed in a file using a digitized scanner. In addition, the sketch is also a reference button to a detailed sketch of a data flow diagram. The detailed data flow diagram is pictured in figure 5.

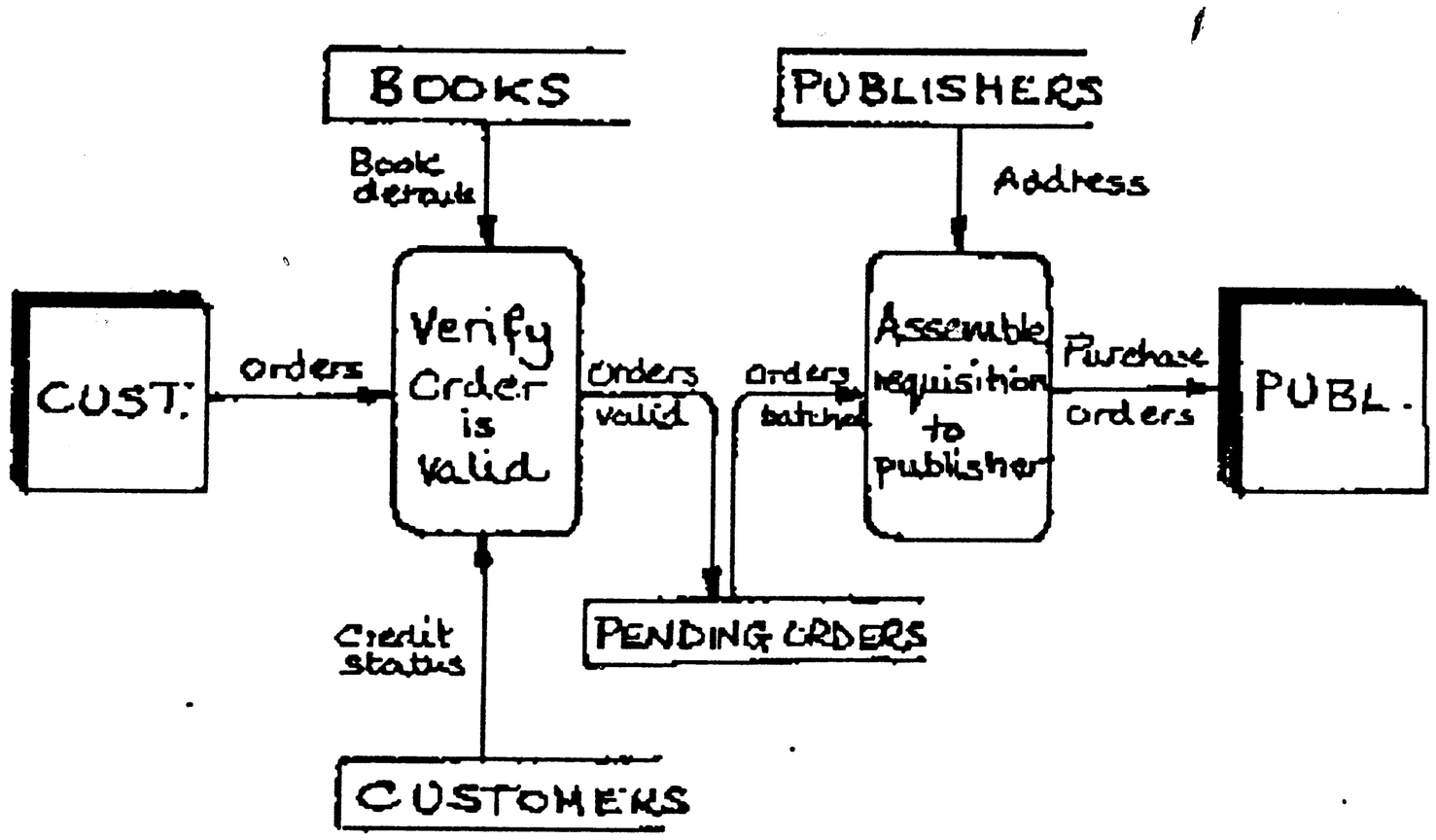
As can be seen in figure 5, the words THIRD NORMAL FORM and DATA STORES are underlined indicating note buttons. The words Books and Customers are defined as reference buttons which are linked to other guidelines. Traversing the Books link, the data dictionary entry for the Books data store is displayed. This is pictured in figure 6.

The word BOOKS in figure 6 is a replacement button. After pushing this button, figure 7 shows the additional information that was hidden.

The two relation names: Books and Author are also replacement buttons. Figure 8 displays the information that was folded behind these two buttons.

Before ending our session, we close all opened guidelines, and before exiting Guide we must close the Guide window.

Expanding the logical Data Flow Diagram highlights the logical functions that make up the proposed system. Illustrated below shows what the data flow looks like now:



By identifying THIRD NORMAL FORM relations as the basic buiding blocks for DATA STORES, the *Books* and *Customers* data stores can be analyzed.

Figure 5. Detailed Data Flow Diagram.

The Data Dictionary entry for books specifies the following structure:

BOOKS

ISBN
- BOOK-TITLE
- AUTHOR
- ORG-AFFILIATION
- PUBLISHER-NAME
- PRICE

Figure 6. The data dictionary entry for the Books data store.

The Data Dictionary entry for books specifies
the following structure:

...Normalization of the BOOKS data store
produces the following relations named:
- o **Books**
- o **Author**

ISBN
BOOK-TITLE
AUTHOR
ORG-AFFILIATION
PUBLISHER-NAME
PRICE

Figure 7. Activating the BOOKS replacement button.

The Data Dictionary entry for books specifies
the following structure:

...Normalization of the BOOKS data store
produces the following relations named:
- o Books
  (ISBN,TITLE,AUTHOR,PUBLISHER-NAME,PRICE)
- o Author
  (AUTHOR,ORG-AFFLIATION)

ISBN
BOOK-TITLE
AUTHOR
ORG-AFFILIATION
PUBLISHER-NAME
PRICE

Figure 8. Activating the Books and Author replacement buttons.

Chapter VI

Conclusion.

The goal of this study has been to gain a better understanding of hypertext, investigate its strengths and weaknesses, and evaluate its effectiveness as an organizational tool during the software development process.

It should be clear from the previous example that the structure of hypertext provides a rich environment for a software designer to operate within. The environment is open ended so that information can be inserted or removed from the structure in a very flexible and easy manner.

The traversal functions provided by hypertext facilitate easy navigation through the environment in a user friendly manner. To visit any piece of information, one merely points and clicks a mouse button.

The purpose of any SDLC documentation base is to guide and direct the software development effort. Many techniques have been proven practical for documenting system requirements and completing design activity.

Graphic-based techniques such as data-flow diagrams (DFDs), entity-relationship diagrams (E/R models), and system structure charts (SSCs) have greatly improved the communication between the system designer and the end user. These techniques promote understanding of the business problem that leads to systems which both meet the user's needs and are easy to maintain.

Supporting this philosphy and conspicuously absent from most software development environments are organizational tools to neatly package this information. Software project organization is represented by life cycle documentation, therefore a tool that organizes, directs, and controls the information base produced by the project should improve software development productivity. Unfortunately, few CASE (Computer-Aided Software Engineering) tools were conceived to support consistent and comprehensive integration of SDLC documentation.

The objective of this research was to provide a perspective on the support functionality of hypertext. The structuring functionality is rich, permitting powerful and creative structures to be built in a very user friendly fashion. The basic idea of hypertext is that ideas correspond to objects and one manipulates

ideas and their relationships by directly manipulating windows and icons.

Hypertext, far from being and end to itself, is really only a step toward the time when the computer is a direct and powerful extension of the human mind, just as Vannevar Bush envisioned when he introduced the concept of Memex.

REFERENCES

Blair & Maron 85

Blair, D. C. and M. E. Maron, "An Evaluation of Retrieval Effectivenes for a Full-text Document-Retrieval System", CACM, Vol. 28, no 3, pp. 289-299, March 1985.

Brown 83

Brown, J. S., "Process versus Product: A Perspective on Tools for Communal and Informal Electronic Learning," in Education in the Electronic Age: A Report from the Learning Lab, WNET/Thirteen Learning Lab, New York, pp. 41-58, 1983.

Brown 86

Brown, P.J., "Interactive Documentation", Software Practice and Experience, Vol. 16, pp. 291-299, March 1986.

Bush 45

Bush, Vannevar, "As We May Think", Atlantic Monthly, no. 176, pp. 101-108, July 1945.

Byers 87

Byers, T.J., "Built by Association", PC World, pp. 244-251, April 1987.

Conklin 87

Conklin, Jeff, "Hypertext: An Introduction and Survey", Computer, pp. 17-41, September 1987.

Delisle 86

Delisle, N., "Neptune: A Hypertext System for CAD Applications", Proceedings of ACM SIGMOD International Conference on Management of Data, Washington D.C., pp. 132-143, May 1986.

Delisle & Schwartz 86

Delisle, Norman and Mayer Schwartz, "Contexts - A Partitioning Concept for Hypertext", proceedings of the MCC Conference on Computer-Supported Cooperative Work, MCC, Austin Texas pp. 147-152, December 1986.

REFERENCES

Lee 87

Lee, W., "?: A Context-Sensitive Help System based on Hypertext", 24th ACM/IEEE Conference Proceedings, pp. 429-434, 1987.

Engelbart & English 68

Englbart, D. C. and W. K. English, "A Research Center for Augmenting Human Intellect", in AFIPS Conference Proceedings, Vol. 33, Part 1, The Thompson Book Company, Washington D.C., 1968.

Garret, Smith & Meyrowitz 86

Garret, Nancy L., Karen E. Smith, and Norman Meyrowitz, "Intermedia: Issues, Strategies, and Tactics in the Design of a Hypermedia Document System", in Proceedings of the Conference on Computer-Supported Cooperative Work, MCC Software Technolgy Program, Austin Texas, 1986.

Goodman 87

Goodman, Danny, The Complete HyperCard Handbook, 720 pages, Bantam, September 1987.

Gould & Grischkowski 84

Gould, John and Grischlowsky, Nancy, "Doing the same work on hardcopy and with Cathode Ray Tube (CRT) Terminals", IBM Research Report RC 9849, January 1983.

Guide User's Manual

Guide User's Manual,
Owl International Inc.,
14218 NE 21st Street,
Bellvue, WA. 98007.

Halasz et al 87

Halasz, F.G., Moran, T.P., & Trigg, R.H., "NoteCards in a Nutshell", Proceedings of the ACM Conference on Human Factors in Computing Systems, Toronto, Canada, April 1987.

Hershey 84

Hershey, William, "ThinkTank", BYTE, p. 189, May 1984.

REFERENCES

Hershey 85

Hershey, William, "Idea Processors", BYTE, pp. 337-350, June 1985.

Malone et al 87

Malone, Thomas W., Kenneth R. Grant, Franklyn A. Turbak, Stephen A. Brobst, and Micheal D. Cohen, "Intelligent Information-Sharing Systems", Communications of the ACM, Vol. 30, No 5, pp. 390-402, May 1987.

Marchionini 88

Marchionini, Gary, and Ben Shneiderman, "Finding Facts vs. Browsing Knowledge in Hypertext Systems", Computer, pp. 70-80, January 1988.

Nelson 80

Nelson, T.H., "Replacing the Printed Word: A Complete Literary System", IFIP Proceedings, pp. 1013-1023, Ocytober 1980.

Nelson 81

Nelson, T.H., "Literary Machines: The Report on, and of, Project Xanadu", 1981.

Por 87

Por, George, "Hypermedia and Higher Education", Computer Currents, pp. 14-16, August 1987.

Shapiro 87

Shapiro, Ezra, "Moving Toward AI", BYTE, pp. 263-266, August 1987.

Shasha 86

Shasha, Dennis, "When Does Non-Linear Text Help", Expert Database Systems, Proceedings of the First International Conference, pp. 109-121, April 1986.

Shneiderman & Morariu 86

Shneiderman, Ben, and Janis Morariu, "The Interactive Encyclopedia System (TIES)", Department of Computer Science, University of Maryland, College Park, MD, June 1986.

Woodhull 86

Woodhull, Albert S., "Kamas", BYTE, p. 241, April 1986.

REFERENCES

Yankelovich, Meyrowitz, van Dam 85
Yankelovich, Nicole, Norman Meyrowitz, and Andries van Dam, "Read and Writing the Electronic Book", Computer, pp. 15-30, October 1985.

Yankelovich 87
Yankelovich, Nicole, "Designing Hypermedia Ideabases - The Intermedia Experience", IRIS Technical Report 87-4, 1987.

Yankelovich et al 88
Yankelovich, Nicole, Bernard J. Haan, Norman K. Meyrowitz, and Steven M. Druker, "Intermedia: The Concept and the Construction of a Seamless Information Environment", Computer, pp. 80-96, January 1988.

Young 86
Young, Jeffery S., "Hypermedia", in Macworld, March 1986.

Personal Data.

Timothy B. Halton was born on April 27, 1957 to Mr. and Mrs. Bernard R. Halton of Minneapolis, Minnesota. After graduating from Cooper Senior High School in June 1975 he was accepted by St. Cloud State University, St. Cloud, Minnesota.

During his freshman year he met his wife Patrice. They were married February 26, 1977. Upon completion of his sophomore year, he transferred to Mansfield State University, Mansfield, Pennsylvania. He graduated with high honors in May 1979 with a Bachelor of Science degreee in Information Processing.

Currently he encourages, directs and supports the End User Computing activities for AT&T Microelectronics.